SIGNATURE VERIFICATION AND AUTHENTICATION METHOD

FIELD OF THE INVENTION

The present invention relates to a method that makes it possible to increase the efficiency, in terms of the calculation time and the RAM and ROM required, of the verification of a signature or an asymmetric authentication requiring several modulo-n or large-number multiplications.

The RSA and Rabin signature or authentication algorithms are examples that allow the implementation of this method.

The method is more particularly adapted to an implementation in the case of a computer, for example a personal computer designated PC, that generates a signature or an authentication by means of a secret key, which must then be verified by a microcomputer card. The microcomputer performs this verification by means of a public key. It has relatively little power compared to the PC.

The term "*microcomputer card*" is intended to mean a standard monolithic microcontroller with an incorporated memory.

The majority of public key algorithms used in the world today perform "*large-number*" modulo calculations. "*Large-number*" designates positive whole numbers of at least 320 bits. For security reasons, the scientific community currently recommends the use of numbers of at least 512 bits, or even 1024 bits for most of the algorithms, for example for the RSA or Rabin algorithms.

Currently, microcomputer cards are brought to dialog with computers having computing capacities much larger than their own. Moreover, for cost reasons, microcomputer cards without an arithmetic coprocessor and with very limited memory resources (ROM, RAM, EEPROM) are used. For this reason, the calculations normally required to perform an authentication verification or a public-key signature verification using large-number modulo calculations are often very long, or even impossible without enough memory, if the traditional descriptions of the cryptographic algorithms are used.

In the description below, the following terms mean:

- "*prover*": the entity that wishes to be authenticated, or that produces a signature. To do this, it performs calculations involving the secret key of the asymmetric algorithm used. It could be, for example, a computer of the PC type.

- "*verifier*": the entity that verifies the authentication, or that verifies the validity of a signature. To do this, it performs calculations involving only the public key of the asymmetric cryptographic algorithm used. It can be, for example, a microcomputer card.

Ins A3

The object of the present invention is to implement a method for verifying signatures and authentications that makes it possible to eliminate the aforementioned disadvantages inherent in the more limited computing capacity of a verifying entity constituted by a microcomputer card, as compared to a proving entity such as a personal computer or the like equipped with a card reading device.

Consequently, another object of the present invention is to simplify the verifier's operations for calculating certain modular reductions through the implementation of additional calculations by the prover, the verifier's task thus being simplified without any reduction in the theoretical security of the system.

The method for verifying a signature, or respectively an authentication, by means of an asymmetric private-key and public-key cryptographic calculation process, which is the subject of the present invention, this method being implemented between a "*prover*" entity and a "*verifier*" entity, the prover entity performing cryptographic calculations with the private key in order to produce a signature calculation, or respectively an authentication value, and the verifier entity, based on this transmitted value, performing cryptographic calculations with this public key in order to perform this signature verification, or respectively this authentication, the cryptographic calculation operations implementing the calculation of modulo n or large-number multiplications, is remarkable in that for a cryptographic calculation process using a public key constituted by a public exponent e and a public modulo n, and a private key constituted by a private exponent d, this method consists of calculating, at the level of the prover entity, at least one prevalidation value and transmitting from the prover entity to the verifier entity this at least one prevalidation value, thereby allowing the verifier entity to perform at least one modular reduction without any division operation for this modular reduction.

The method that is the subject of the present invention applies to any dialogue or protocol for exchanging messages between a prover entity such as a personal computer and a verifier entity such as a microcomputer card, particularly in connection with banking transactions, access control, or the like.

BRIEF DESCRIPTION OF THE DRAWINGS

It will be more clearly understood by reading the description below and examining the drawings, in which:

- Fig. 1 represents a diagram illustrating the method that is the subject of the present invention, implemented between a prover entity and a verifier entity;

- Fig. 2a represents a diagram illustrating the method that is the subject of the present invention, implemented with a Rabin authentication verification algorithm;

- Fig. 2b represents a diagram illustrating the method that is the subject of the present invention, implemented with a Rabin signature verification algorithm;

- Fig. 3a represents a diagram illustrating the method that is the subject of the present invention, implemented with an RSA authentication verification algorithm;

- Fig. 3b represents a diagram illustrating the method that is the subject of the present invention, implemented with an RSA signature verification algorithm.

A more detailed description of the method that is the subject of the invention is given in connection with Fig 1 and the subsequent figures.

The method that is the subject of the invention implements, at the verifier entity level, public-key algorithms requiring modulo-n or large-number multiplications, and modifies them slightly by having one or more quotients q calculated externally, i.e. at the prover entity level, and by supplying this quotient or quotients to the verifier. Thus, the verifier can more easily and quickly calculate certain modular multiplications: instead of calculating a*b modulo n, it only has to calculate a*b, q*n, and a*b-q*n, a and b designating values of the signature or authentication verification calculation. Sometimes, for security reasons,, it uses the latter value in a way that allows it to make sure that this latter value is actually between 1 and n. When an algorithm is thus modified by "*precalculating*" certain quotients that are supplied to the verifier in order to simplify the calculations executed by the latter, it is called a "*subjacent*" algorithm in order to designate the initial algorithm from which it is derived, prior to performing this modification. Thus, in reference to Fig. 1, according to a remarkable aspect of the method that is the subject of the present invention, the quotient or quotients q that verify the relation q=a*b/n constitute one or more prevalidation values transmitted to the verifier entity in order to allow the verifier entity to perform at least one modular reduction without any division operation for this modular reduction. Referring to Fig. 1, it is indicated that the method that is the subject of the invention can be implemented either when verifying the authentication after the sending of an

prompt value such as a random value a (see the reference 0 in the figure), the internal calculation (reference 1) at the prover level of a response value $b = a^d \bmod n$ and the prevalidation value q, the transmission (reference 2) of b and q from the prover to the verifier, and the calculation (reference 3) by the verifier of the quantities a*b, q*n and a*b-q*n in order to perform the verification of the authentication, or when verifying the signature of a message M after the calculation (reference 1) at the prover level of a signature $S = S_d(M)$ for the message M and the prevalidation value q, the sending (reference 2) of q, S and M from the verifier to the prover, the calculation (reference 3) at the verifier level of the quantities a*b = S*S, q*n and a*b-q*n in order to perform the signature verification.

In Fig. 1 and the subsequent figures, a straight arrow represents the transmission of the aforementioned values from the verifier to the prover or vice versa, and a looped arrow at the prover level or the verifier level represents the implementation of an internal calculation at the prover level or the verifier level. Finally, in the description below, the response R designates either the value b calculated by encrypting the random number a in the case of an authentication verification $b = a^d \bmod n$, or the signature value $S = S_d(M)$ following the connection of the verifier and the prover.

Various examples of the implementation of the method that is the subject of the present invention will now be described based on subjacent algorithms, designated by RSA and Rabin algorithms.

Subjacent RSA and Rabin algorithms

The RSA algorithm is the most famous of the asymmetric cryptographic algorithms. It was invented by RIVEST, SHAMIR and ADLEMAN in 1978. Its description may be found in:

R. L. Rivest, A. Shamir, L.M. Adleman: "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM*, 21, No. 2, 1978, pp. 120-126, or in the following documents:

- ISO/IEC 9594-8/ITU-T X.509, Information Technology – Open Systems Interconnection – The Directory: Authentication Framework;
- ANSI X9/31-1, American National Standard, Public-Key Cryptography Using Reversible Algorithms for the Financial Services Industry, 1993.

These documents are introduced into the present description as references.

The RSA algorithm uses a whole number n that is the product of two large prime numbers p and r, and a whole prime number e with ppcm(p-1, r-1), and such that $e \bullet \pm 1$ modulo ppcm(p-1 ,r-1). The integers n and e constitute the public key. The public key calculation uses the function α of $\mathbf{Z}/n\mathbf{Z}$ in $\mathbf{Z}/n\mathbf{Z}$ defined by $\alpha(x)=X^e$ mod n. The secret key calculation uses the function $\alpha^{-1}(y)=y^d$ mod n, where d is the secret exponent, also called the "*secret key*" or "*private key*," defined by $ed \equiv 1$ mod ppcm(p-1, r-1).

Let n be the RSA public modulo, let d be the RSA secret exponent and let e be the RSA public exponent.

In the case of an authentication verification, the verifier generates a random number A modulo n, and sends it to the prover. The latter then calculates $B= A^d$ modulo n, and returns this value B to the verifier. The latter accepts the authentication if and only if B^e modulo n = A.

The smallest value of e for using the RSA algorithm is e = 3. For e = 2, the Rabin algorithm is used; the latter will be described later in the description. This value e = 3 is advantageous since it allows the verifier to have only two modular multiplications to perform.

The Rabin algorithm is similar to an RSA algorithm with the public exponent e = 2. In fact, when e = 2, the function x^e is not bijective modulo n, when n is the product of two prime numbers > 2, so slight modifications are introduced in the use of the Rabin algorithm as compared to the RSA algorithm.

A description of the Rabin algorithm may be found in:

M.O. Rabin, "Digitized Signatures and Public-Key Functions as Intractable as Factorization," Technical Report LCS/TR-212, M.I.T. Laboratory for Computer Science, 1979, introduced in the present patent application as a reference.

Exemplary implementations of the method that is the subject of the invention using the Rabin and RSA algorithms

♦ Rabin algorithm

The method that is the subject of the present invention will first be described in a particular non-limiting embodiment based on the Rabin algorithm, or for e = 2.

♦♦ Authentication verification

As represented in Fig. 2a, a possible example of the utilization of the Rabin algorithm in authentication verification is described below.

Let n be the public modulo. The verifier generates a random number A modulo n, and sends it (reference 0 in the figure) to the prover. The latter then calculates a number B (reference 1), and sends this value B to the verifier. The latter accepts the authentication if and only if B*B modulo n is equal to one of the following four possible values: A, or n-A, or C*A modulo n, or – C*A modulo n. C is a number set by the protocol, most often C = 2.

In order to simplify the verification process in accordance with the method that is the subject of the present invention, the prover does not send (reference 2) the value B alone: it sends B and Q, where Q is the quotient of B*B by the public modulo n. The verifier then verifies that D_{AR} = B*B = Q*n is actually equal to one of the following four values: A, n-A, (C*A) modulo n, or (-C*A) modulo n. In addition, it can calculate (C*A) modulo n, by calculating C*A, keeping this value if it is < n, and otherwise taking the value C*A – n. Thus, the verifier does not have any division to perform.

♦♦ Signature verification

Thus, as represented in Fig. 2b, and keeping the same notations as above, let M be the message whose signature S the verifier wishes to verify. The signature S is obtained from the private key d by S = S_d(M), S_d(M) designating the operation for calculating the signature of the message M. If S is a Rabin signature of M, then the verifier normally verifies that S*S modulo n = f(M) or n-f(M), or (2*f(M) modulo n) or (-2*f(M) modulo n), where f is a standardized public function of the message M. For example, f is the identity function, or is described in a signature standard; for example, it is possible o use the *padding* or concatenation operations of the PKCS#1 standard normally established for RSA; see the descriptive elements of this standard later in the description.

Keeping the same notations as above, in order to simplify the signature verification process as represented in Fig. 2b, in the method that is the subject of the present invention, the prover does not send (reference 2) the value S alone: it sends S and Q, where Q is the quotient of S*S by the public modulo n. The verifier then verifies that D_{SR} = S*S - Q*n is actually equal to f(M), or n-f(M), or C*f(M) modulo n, or –C*f(M) modulo n, where C is a number set by the protocol, C being able to be taken as equal to 2. Since these last two values can be modulo-n calculated by performing zero or ~~a~~ subtraction by n, the verifier no longer has any division to calculate.

♦ RSA algorithm

The method that is the subject of the present invention will now be described in a particular non-limiting embodiment based on the RSA algorithm, or for e = 3.

♦♦ Authentication verification

As represented in Fig. 3a, beginning with a random number A, in order to simplify the verification process, in the present invention, the prover does not send (reference 2) the value B alone: it sends B, Q1 and Q2, where Q1 is the quotient of B*B by the public modulo n, and where Q2 is the quotient of B* (B*B – Q1*n) by n. The verifier then verifies that D_{ARSA} = B*(B*B – Q1*n) –Q2*n is actually equal to A. Thus, the verifier no longer has any division to perform.

♦♦ Signature verification

Keeping the same notations as above and letting M be the message whose signature S the verifier wishes to verify, S is an RSA signature of M, so the verifier normally verifies that S^e modulo n = f(M), where f is a standardized public function of the message M. For example, f is the identity function, or is described in an RSA signature standard, such as for example the PKCS#1 standard. The standardized public function can consist of applying a condensation function SHA-1 to the message M in order to obtain a message digest CM, then of concatenating this message digest with a constant value.

Thus, as represented in Fig. 3b, and keeping the same notations as above, in order to simplify the signature verification process, in the method that is the subject of the present invention, the prover does not send (reference 2) the value S alone: it sends S, Q1 and Q2, where Q1 is the quotient of S*S by the public modulo n, and where Q2 is the quotient of S*(S*S – Q1*n) by n. The verifier then verifies that D_{SRSA} = S*(S*S = Q1*n) –Q2n is actually equal to f(M). Thus, the verifier no longer has any division to perform.

The condensation function SHA-1 is a public "*condensation*" function. It takes as input a message whose size can run from 0 bytes to several gigabytes, and yields as output a 160-bit "*digest*" of the message. This function is often used in standards or with signature algorithms, since it is reputed to be collision-resistant, which means that it is not known how to concretely find two separate messages that have the same message digest (they exist, but it is not known how to find such a pair of messages). This makes it possible to sign the message digests rather than the messages themselves.

The PKCS#1 standard is an RSA signature standard. It describes a public function f. This function f is applied to the message M to be signed with RSA before launching the RSA modular exponentiation operation itself: the RSA signature of M is therefore $S = (f(M))^d$ modulo n, where is the RSA public modulo and where D is the RSA secret exponent. f uses a condensation function (for example SHA-1) followed by a *padding*, or concatenation, with a constant.

For a more detailed conscription, please consult:

PKCS#1, *RSA Encryption Standard*, Version 2, 1998, available at the following address:

ftp://ftp.rsa/com/pub/pkcs/doc/pkcs-1v2.doc

whose published version is introduced in the present application as a reference.

The invention thus consists of supplying additional data to the verifier in order to facilitate its calculations. In order to precalculate this data, in this case the quotients constituting the prevalidation value or values, it is not necessary to use the secret key of the algorithm. This means that this data is completely redundant relative to the values transmitted to the card in a "*conventional*" utilization of the asymmetric algorithm. In fact, in the "conventional" version, the card knows how to find these quotients itself. There is therefore no additional information supplied to the card, in the sense of information theory, when the method that is the subject of the present invention is implemented as described above. This shows that the security of the system is in no way weakened as compared to the "*conventional*" implementation of the algorithm.

ins Q5